Adam Thorne

440-731-7857 | at@adamthorne.com | linkedin.com/in/adam-thorne-55b931261 | github.com/adamthorne27 | adamthorne.com

Education

University of Michigan Ann Arbor, MI
B.S. Computer Science, B.S. Applied Mathematics *Expected May 2028*
Transfer: Transferred from University of California, Davis; Prior GPA: 3.98

Relevant Coursework: Data Structures, Algorithm Analysis, Object-Oriented Programming, Operating Systems, Computer Architecture, Database Systems, Probability, Linear Algebra

Experience

Machine Learning Analyst Intern Summer 2026
Lightshift Energy
- Expanded production-adjacent peak-load forecasting research across 3 operating regions by using AI-assisted code tracing and test scaffolding, enabling safe iteration across Python, notebook, and GPU-training workflows.
- Cut model-training time 3.5x by profiling GPU/data bottlenecks, removing redundant memory clears, and switching checkpoints to safetensors with mixed precision, reducing epoch time from 30.3s to 8.6s with no accuracy loss.
- Reduced evaluation dependency risk by building a 900-line, 9/9-tested offline scoring library, reproducing production peak-demand metrics without cloud-warehouse or hardcoded-date dependencies.
- Reduced a regional data-pipeline stall from a projected 12+ hours to 15–20 minutes by replacing full-table lookups with batched PyArrow scans, sparse caching, and fork-safety fixes, cutting memory from roughly 53 GB/worker to under 1 GB/worker.
- Scaled research throughput by automating architecture sweeps, loss-function experiments, weather encoders, HPO trials, and validation reports, training and scoring roughly 1,800 configurations over about 800 GPU-hours.

Founding Machine Learning Engineer and Researcher Feb 2026 – Present
Verazoi
- Produced reusable artifacts for 45-user health forecasting experiments by transforming glucose, meal, activity, heart-rate, and bio data into feature schemas, split summaries, and evaluation outputs.
- Standardized model comparison across balanced accuracy, F1, AUPRC, recall, false alerts/day, and lead time by building evaluation tooling with MLflow tracking and user-bio ingestion.
- Helped secure USD 2,000 in pitch funding by explaining model risks, validation design, user impact, and technical roadmap tradeoffs to a nontechnical judging panel.

Project Manager & Machine Learning Engineer Oct 2025 – Present
Machine Learning Student Network
- Enabled reproducible notebook-first ML workflows for a multi-person quant team by building a Python research platform with Parquet data caching, TOML configs, validation tests, QuantStats reports, shared MLflow, and GitHub/Colab workflows.
- Improved contributor reliability by defining baseline templates, reproducible configs, model-output contracts, and review standards for ridge, boosting, neural, autoencoder, and sequence models.

Projects

C++ Concurrent Event Processing Engine | *C++20, Linux, Ring Buffers, Atomics, CMake*
- Processed 4.2 million event messages per second with p99 latency under 35 microseconds by implementing cache-friendly state, preallocated storage, and lock-free producer-consumer handoff.
- Compared SPSC and MPSC ring-buffer paths against mutex-backed queues by benchmarking throughput, backpressure, cache locality, and tail-latency behavior under seeded workloads.
- Kept correctness reproducible with deterministic replay, invariant checks, sanitizer runs, CTest/gtest coverage, and benchmark reports that separate warmup, median, and p99 behavior.

Vectorized Columnar Analytics Engine | *C++20, SQL, Arrow, CMake, Google Benchmark*
- Built a columnar execution engine for scans, filters, projections, and hash aggregation over typed batches using vectorized operators and memory-aware iterators.
- Validated query semantics with row-engine comparisons, DuckDB checks, generated datasets, null-handling tests, aggregation invariants, and reproducible benchmark scripts.

Batch ML Feature and Inference Platform | *Python, PyTorch, DuckDB, MLflow, Docker*
- Built a reproducible batch-inference platform with feature materialization, model-version registry, partitioned prediction outputs, drift checks, and historical replay over time-series data.

Distributed Task Queue | *Go, PostgreSQL, Docker, Prometheus*
- Modeled reliable asynchronous processing under backend load by implementing worker leases, retry semantics, dead-letter handling, idempotent handlers, crash-recovery tests, and job-level metrics.

Skills

Languages: Python, C/C++, SQL, TypeScript, JavaScript, Rust, Go
Backend / Infra: Linux/Unix, Git, Docker, FastAPI, PostgreSQL, Redis, APIs, caching, CI, data modeling
Testing / Reliability: pytest, gtest, profiling, benchmarking, concurrency, ThreadSanitizer, GDB, perf
Data / ML: PyTorch, NumPy, Pandas, Polars, DuckDB, PyArrow, Parquet, MLflow, safetensors